Review Seminararbeit

Seminar Software Engineering für verteilte Systeme

Daniel Sturm (1453079)

SoSe 2024

Hinweise

- Länge des Reviews: 2–3 Seiten (inklusive dieser Hinweise)
- Jede Frage in diesem Template muss beantwortet werden! **Ersetzen** Sie dazu die vorhandenen `\todo`-Befehle im Template durch Ihre Antworten in Fließtext oder ausführlichen Stichpunkten.
- Die Qualität der von Ihnen verfassten Reviews geht in Ihre Gesamtnote für das Seminar ein.
- Entfernen Sie nach dem vollständigen Bearbeiten des Reviews das `todonotes`-Paket im Header dieser Datei. Wenn Sie alle `\todo`-Befehle ersetzt haben, kompiliert das Dokument weiterhin ohne Fehler. `\usepackage[color=smdsblue!25]todonotes`

Allgemeine Informationen

Titel der zu bewertenden Arbeit

State of the Art: Compiler-based Static Code Analysis

Hauptinhalt der Arbeit

Zu Beginn der Arbeit wird die statische Codeanalyse als Mittel zur Verbesserung der Codequalität vorgestellt. Außerdem wird der Aufbau der Arbeit beschrieben.

Im Kapitel „Softwaretesting“ wird aufgezeigt, dass es oft zu aufwendig ist, komplexe Software zu testen und dass man den Testumfang meist proportional zum möglichen Risiko wählt. Im Folgenden wird die statische Codeanalyse als Whitebox-Verfahren vorgestellt, bei dem der Code nicht ausgeführt wird. Bei der statischen Codeanalyse wird der Code neben der Syntax auch oft auf Programmierguidelines geprüft. Neben der manuellen Prüfung gibt es auch Tools, die einen Analysebericht erstellen. Hierbei wird auch der Unterschied zwischen falschen positiven und falschen negativen Fehlern hervorgehoben. Der Compiler lässt sich hierbei auch als statischer Codeanalysator betrachten.

Im darauffolgenden Kapitel wird die statische Codeanalyse anhand eines C-Code-Beispiels mit typischen Fehlern vorgestellt. Außerdem werden die Datenflussanalyse und die Kontrollflussanalyse vorgestellt. Bei der Datenflussanalyse betrachtet man die Zustände der Variablen. Insbesondere wird erläutert, dass beispielsweise eine du-Anomalie, bei der eine Variable definiert, aber ihr kein Wert zugewiesen wird, nicht unbedingt zu einem Fehler führt, wenn die Variable nicht ausgelesen wird. Die Kontrollflussanalyse analysiert, wie der Name schon sagt, den Kontrollfluss des

Programms. Dieser wird oft als Graph visualisiert. In einem Beispiel wird anhand eines solchen Graphen die zyklomatische Zahl eingeführt, welche ein Maß für die Komplexität des Codes ist.

Weiter werden in der Arbeit FindBugs als ein Tool zur statischen Codeanalyse für Java, Snyk als KI-basiertes Tool, um Schwachstellen im Code zu finden, und SonarQube als Tool, welches Rückmeldung beim Coden gibt, vorgestellt.

Im Kapitel „Herausforderungen“ wird noch einmal der hohe Aufwand der statischen Codeanalyse betont. Es wird darauf hingewiesen, dass trotz statischer Codeanalyse, z.B. für Logikfehler, noch die dynamische Codeanalyse benötigt wird und man auch trotz Tools immer noch auf falsche positive und falsche negative Fehler sowie auf deren Kommentierung achten muss.

Zum Schluss wird ein Ausblick in die Zukunft gegeben, in welchem die Fehler der statischen Codeanalyse durch KI-Tools analysiert werden können. Allgemein wird festgehalten, dass die Codequalität durch die statische Codeanalyse gesteigert wird und früh gefundene Fehler Kosten sparen.

Allgemeine Bewertung

Stärken der Arbeit

- Klar strukturierter Aufbau: Einleitung, Grundlagen, praktische Beispiele, Herausforderungen, Ausblick
- Gute Erklärung von falsch positiven und falsch negativen Fehlern
- Praktische Beispiele im Kapitel „Softwaretesting“

Schwächen der Arbeit

- Kapitel 3
  - Tools FindBugs, SonarQube und Snyk werden nur oberflächlich behandelt
    * Was genau ist Snyk? Name des Unternehmens? Welches Tool von Snyk soll genau betrachtet werden?
  - Einzelne Tools genauer vorstellen
    * Eventuell Anwendungsbeispiele eines oder mehrerer Tools
  - Wie verwenden die Tools die Datenflussanalyse und Kontrollflussanalyse?
- Kapitel 4
  - Beispiel für hohen Aufwand geben
  - Ist neben der Anzahl und Auswertung der Fehler nicht auch die Rechenzeit der Analyse bei komplexen Programmen relevant?
- Zukunftsausblick
  - KI-Tool? Welches genau? Quelle?

Nutzung KI-basierter Tools

Ich denke anhand der plausiblen Quellenangaben der Arbeit und Prüfung mittels eines Tools zur Erkennung von KI-generierten Texten, dass in dieser Arbeit keine KI-Tools verwendet wurden.

Sachliche Korrektheit

Die verwendeten Quellen sind plausibel, allerdings könnte die ein oder andere Aussage mit zusätzlichen Quellen belegt werden. Siehe dazu z.B. den Punkt Zukunftsausblick in den Schwächen der Arbeit.

Äußere Form

- Die Rechtschreibung und Grammatik ist insgesamt gut. Jedoch könnte man ein paar Kleinigkeiten verbessern.
  - z.B. könnte man in der Zusammenfassung bei compilerbasierten statischen Codeanalyse einen Bindestrich einfügen, um konsistent zum englischen Titel zu sein.
  - Bei der Quellenangabe des Codebeispiels auf Seite 3 passt die Formatierung der Quellenangabe nicht ganz (Klammerfehler)
- Die Kapiteleinteilung ist sinnvoll und nachvollziehbar. Das Grundlagenkapitel könnte jedoch im Weitern Verlauf der Arbeit weiter aufgegriffen werden. Dazu könnte man z.B. zeigen, wie die Datenflussanalyse und Kontrollflussanalyse in den Tools verwendet werden.
- Es wird eine angemessene Anzahl an Abbildungen verwendet, welche korrekt beschriftet sind und im Fließtext referenziert werden.
- Meiner Meinung nach sind die Zeilenumbrüche etwas zu weit auseinander.